



Sécurité, gestion des droits utilisateurs, rôle et lien avec la gestion des utilisateurs externes

Auteurs :

Robin Guerard

Nicolas Lépinay

Matéo Palmieri

Injin Kim

Virgil Nauleau

Cyprien Siaud

Table des matières

Table des matières.....	2
Introduction	3
Les vulnérabilités de MySQL	4
1. Les vulnérabilités antérieures (DoS)	4
2. Les injections SQL	5
Comprendre les enjeux de sécurité	8
Installation sécurisée de MySQL	9
Sécurité du système d'exploitation	12
Sécurité du système Linux	13
Sécurité des mots de passe	14
Sécurité du système de fichiers	16
Protection de la base de données	18
Types de droits existants	19
Gestion des utilisateurs lors de la création de routines	22
Création, suppression, modification des comptes utilisateurs	24
Gestion des privilèges et des droits	24
Sécurité du réseau	26
Connection Over SSL	32
Authentification externe	33
Sources	34

Introduction

La popularité de MySQL en tant que SGBD open-source l'a rendu incontournable dans le monde de la data. Il est utilisé par certaines des applications web les plus utilisées comme Facebook, Twitter, Youtube et WordPress. Avec une telle profusion de données et d'applications clientes différentes, la sécurisation des données devient un enjeu majeur.

Dans ce document, nous présenterons les principaux risques sécuritaires auxquels est exposé le système MySQL, ainsi que les principes de précaution à respecter pour garantir la sécurité des bases de données MySQL.

Les vulnérabilités du système MySQL

MySQL présente des vulnérabilités qui lui sont propres et qu'il faut prendre en considération pour maintenir la sécurité des données. Dans cette partie, nous détaillerons la nature de ces vulnérabilités et leurs solutions possibles.

1. Vulnérabilités antérieures

Certaines vulnérabilités sont toujours d'actualité, tandis que d'autres ont été patchées dans certaines mises à jour. C'est le cas, par exemple, de **l'attaque par déni de service**, qui a été corrigée dans la version 5.5 de MySQL, mais qui affectent toujours les versions antérieures.

Les attaques DoS comptent parmi les cyberattaques les plus répandues et les plus dangereuses dirigées contre les serveurs web ou les instances de serveur comme MySQL.

L'objectif est de rendre le service temporairement ou définitivement indisponible, et d'impacter le système global auquel l'instance est connectée.

Dans le cas de MySQL, le but d'une telle attaque serait de faire crash l'instance MySQL et de la rendre inutilisable par les autres services qui y sont connectés pour leur base de données.

Cette vulnérabilité permet aux utilisateurs authentifiés de saisir une commande SELECT suivie par une commande UpdateXML et une portion de XML contenant un nombre élevé d'éléments imbriqués. Par exemple :

```
SELECT UpdateXML('<a>$a<b>ccc</b><d></d></a>', '/a', '<e>fff</e>') AS val1
```

Cela rend MySQL vulnérable à une attaque DoS.

2. Injections SQL

S'appliquent à :

- SQL Server (toutes les versions prises en charge)
- Azure SQL Database
- Azure SQL Managed Instance
- Azure Synapse Analytics Analytics
- Platform System (PDW)

Le fonctionnement

Les injections SQL → insertions directes de code dans les entrées utilisateur concaténées avec des commandes SQL et exécutées.

Des attaques par injection moins directes insèrent un code malveillant dans les chaînes destinées à être stockées dans une table ou en tant que métadonnées.

Lorsque les chaînes stockées sont ensuite concaténées dans une commande SQL dynamique, le code nuisible est exécuté.

Injection = terminer prématurément une chaîne de texte et ajouter une nouvelle commande.


Terminer la chaîne injectée par une marque de commentaire « -- ». Le texte qui vient à la suite est ignoré au moment de l'exécution.

- « ; » => fin de requête
- « -- » commentaire
- Si le code modifié est syntaxiquement correct, il sera exécuté par le serveur.
- Exécution → Sélection pour OrdersTable où ShipCity est 'Redmond' puis suppression de OrdersTable.


Code syntaxiquement correct ? Indétectable.

 **Vérification des entrées utilisateur.**

Bonnes pratiques

-  Données utilisateurs reçues.
- Prévoir :
 - Insertion fichier JPEG au lieu d'un texte
 - Insertion d'une instruction `DROP TABLE` dans un champ texte
- Tester taille et type des données reçues
- Tester le contenu des variables pour n'accepter que les valeurs attendues.
- Rejeter les entrées contenant des données binaires, des séquences de caractères d'échappement et des caractères de commentaire

Peut empêcher l'injection de scripts et protéger contre l'utilisation de dépassements de mémoire tampon.

- Si docs XML, validation des données basées sur les schémas.
-  `Transact-SQL` dans un Input
- Utilisation de procédures stockées
- Validation des données avant d'entrer dans les zones de confiance
- Plusieurs niveaux de validation :

Utilisateurs malintentionnés \neq pirates déterminés.

Validation dans les Input et à toutes les étapes du programme

La validation des données dans une application côté client peut empêcher l'injection de scripts simples.

Si le niveau suivant considère que son entrée a déjà été validée, un utilisateur malveillant capable de contourner un client peut disposer d'un accès complet à un système.

Pas de concaténation d'une entrée utilisateur \rightarrow point d'entrée principal

Utilisation de paramètres SQL de type sécurisé

- SQL Server possède une collection `Parameters` qui fournit le contrôle du type et valide la longueur.
- Les entrées traitées comme des string pas comme du code exécutable
- Permet des contrôles de type et de longueur.
- Déclenche une exception si la valeur n'est pas dans les limites
-

Utilisation d'entrées paramétrables avec des procédures stockées

Les injections SQL peuvent cibler les procédures stockées notamment si elles utilisent des entrées non filtrées.

Lors de l'utilisation de procédures stockées, il est impératif d'utiliser des paramètres en entrée.

Utilisation de la collection Parameters avec des instructions SQL dynamiques

Si l'utilisation de procédure stockée est impossible, il est quand même possible d'utiliser des paramètres.

Filtrage des entrées

Le filtrage des entrées peut être utile pour éviter des injections SQL par la suppression des caractères d'échappement.

Peu fiable car beaucoup de caractères peuvent poser des problèmes.

Clauses LIKE

Même avec `LIKE`, les caractères génériques devront quand même être séparés par des caractères d'échappement

Examen du code à la recherche d'injection SQL

Examen du code qui appelle `EXECUTE`, `EXEC` ou `sp_executesql`.

Utilisation de requêtes pour identifier des procédures qui contiennent ces instructions.

Recherche de 1, 2, 3 ou 4 espaces après les mots `EXECUTE` ou `EXEC`.

Enveloppement de paramètres avec `QUOTENAME()` et `REPLACE()`

Dans chaque procédure stockée vérification que les variables utilisées dans du Transact-SQL dynamique sont traitées correctement.

Les données issues des paramètres d'entrée de la procédure stockée ou lues à partir d'une table doivent être enveloppées dans QUOTENAME() ou REPLACE().

La valeur de @variable transmise à QUOTENAME() est de type sysname et sa longueur maximale de 128 caractères.

Injection activée par la troncature des données

Une variable Transact-SQL dynamique affectée à une variable est tronquée si elle est supérieure à la mémoire tampon allouée à cette variable.

Un attaquant capable de forcer la troncature en transmettant une chaîne plus longue peut manipuler le résultat.

La procédure stockée créée par le script suivant court un risque d'injection permis par la troncature.

En transmettant 154 caractères dans une mémoire tampon de 128 caractères, un attaquant peut définir un nouveau mot de passe sans connaître l'ancien.

C'est pourquoi il faut utiliser une mémoire tampon volumineuse pour une variable de commande ou pour exécuter le Transact-SQL dans l'instruction EXECUTE.

Comprendre les enjeux liés à la sécurité

Configurer et maintenir la sécurité de ses serveurs MySQL est une composante essentielle de l'administration de bases de données. Dans cette partie, nous expliquerons comment reconnaître les multiples éléments servant à protéger et sécuriser les bases de données.

La sécurité de l'environnement MySQL démarre à l'**installation**. La personne ayant réalisé l'installation, l'emplacement où l'installation a été effectuée, et les privilèges accordés au compte *root* sont des étapes clés qui peuvent être à l'origine de failles de sécurité majeures si elles ont été négligées.

Une fois la base de données créée, il est impératif de définir des **contrôles d'accès** à celle-ci. L'administrateur doit être conscient de tous les objets auxquels les utilisateurs auront accès et réguler ces accès aux moyens de permissions. La difficulté de cette tâche augmente évidemment avec le nombre de bases de données présentes sur le serveur et leurs tailles respectives. Les utilisateurs doivent avoir un accès restreints à certaines bases de données, puis un accès restreints à certaines tables à l'intérieur de ces bases de données.

Enfin, des questions doivent se poser concernant la sécurité globale du **serveur** sur lequel tourne MySQL : comment la connexion au serveur s'effectue-t-elle ? La connexion est-elle câblée ou sans fil ? Qui s'y connecte ? Les utilisateurs doivent-ils s'authentifier ou des visiteurs non-authentifiés sont-ils autorisés dans l'environnement ?

Des restrictions pour MySQL peuvent être configurées dans l'environnement réseau pour ne donner accès qu'au *localhost* ou pour définir un groupe d'hôtes communicants, empêchant ainsi tel serveur de communiquer avec tel autre.

Installer MySQL de manière sécurisée

La sécurité de l'environnement MySQL commence dès l'installation. Sa popularité s'est traduite par un accroissement des vecteurs d'attaque et des exploits informatiques, et modifier la configuration par défaut installée par MySQL est primordial.

Dans cette section, nous détaillerons comment améliorer la sécurité de MySQL immédiatement après son installation.

1. Supprimer la base de données Test

La base de données « test » installée par le package MySQL Server au cours du process `mysql_install_db` est entièrement accessible par tous les utilisateurs par défaut. Cela en fait une cible de choix pour les attaquants. Il est donc conseillé de la supprimer post-installation.

2. Supprimer tous les comptes anonymes

Par défaut, MySQL crée plusieurs comptes utilisateurs qui n'ont aucune utilité après l'installation. Leur présence dans le système donne aux attaquants un point d'entrée dans la base de données. Il est donc recommandé de les supprimer.

3. Modifier les redirections de port par défaut

MySQL tourne sur le port 3306 par défaut. Il est conseillé de changer ce port après l'installation pour ne pas dévoiler quels services tournent sur tel port, puisque les attaquants sont susceptibles de tenter d'exploiter des valeurs par défaut au préalable.

4. Changer les hôtes ayant accès à MySQL

Si l'instance MySQL a été installée comme serveur autonome (c'est notamment le cas si l'application et le serveur web interrogent la base de données depuis un autre serveur), MySQL doit être configurée pour n'autoriser l'accès qu'aux hôtes autorisés. Cela est possible en apportant des modifications aux fichiers `hosts.deny` et `hosts.allow`.

5. Supprimer et désactiver le fichier d'historique MySQL

Tout comme la base de données Test, un fichier d'historique nommé `.mysql_history` est généré par défaut à la racine après l'installation. Il contient des détails d'historique liés aux étapes d'installation et de configuration qui ont été

exécutées. Le leak de ce fichier pourrait entraîner une exposition des mots de passe. Il doit donc être supprimé.

Il est également recommandé de créer un lien symbolique depuis le fichier `.mysql_history` vers le fichier `/dev/null` pour empêcher les logs vers ce fichier.

6. Ne pas lancer MySQL avec les privilèges root

MySQL doit être lancé avec un compte utilisateur spécifique nouvellement créé qui possède les permissions appropriées pour lancer le service, et non pas l'utilisateur root. Cela facilite les audits et donne des informations concernant l'utilisateur dans les logs, et empêche les attaquants de prendre le contrôle en hackant le compte root.

7. Désactiver l'authentification à distance

Si la base de données est uniquement utilisée par des applications locales, l'accès distant au server doit être désactivé. On peut faire ça en modifiant le fichier `/etc/my.cnf`. Configurer MySQL pour qu'il arrête d'écouter tous les ports TCP/IP permettra de restreindre l'accès à la base de données à des communications MySQL locales par sockets.

8. Limiter ou désactiver la commande SHOW DATABASE

Là encore, cette commande renvoie trop d'informations qui pourraient être utiles à un attaquant. Elle peut être restreinte ou désactivée complètement en ajoutant `skip-show-database` dans le fichier `my.cnf`.

9. Désactiver la commande LOAD DATA LOCAL INFILE

Cette commande permet aux utilisateurs de lire des fichiers locaux et d'accéder à d'autres fichiers sur le système d'exploitation, ce qui pourrait être exploité dans le cadre d'une injection SQL. Cette commande doit donc être désactivée en ajoutant `set-variable=local-infile=0` dans le fichier `my.cnf`.

10. Renommer le compte root

Changer le nom du compte root en un nom plus difficile à deviner ajouter une couche de sécurité supplémentaire, puisque les attaquants devront déterminer le nom du compte du super-utilisateur avant de pouvoir brute-forcer son mot de passe.

11. Configurer les bonnes permissions de fichier

S'assurer que le fichier **my.cnf** n'est modifiable que par l'utilisateur root, et que l'accès aux données dans le dossier **/usr/local/mysql/data** est correctement sécurisé.

C'était 11 étapes pour un déploiement de MySQL plus sécurisé.

La sécurité du système d'exploitation

Si le système d'exploitation du serveur sur lequel est hébergé notre base de données n'est pas sécurisé, alors nos données ne le sont pas non plus.

C'est pourquoi il est important de mettre à jour son OS, d'utiliser et mettre à jour son antivirus, de configurer et utiliser un firewall afin de sécuriser et monitorer le trafic du réseau et ne laisser aux comptes connectés que les accès nécessaires pour leur travail (principe du moindre privilège).

Enfin il faut également utiliser le système de permission du système d'exploitation utilisé afin de restreindre l'accès aux seuls dossiers et fichiers utiles pour chaque compte.

L'objectif de toutes ces manipulations est d'avoir un contrôle total sur ce qui se passe sur le serveur, qui se connecte, à quoi a-t-il accès, qu'à t'il fait, quand et comment... Et ainsi pouvoir agir selon ce qui a été fait, gérer des droits d'accès qui doivent être révoqués ou au contraire augmentés.

Protections des données du système Linux

Pourquoi Linux est-il plus sécurisé que la concurrence (Windows) ?

Le format des fichiers sous Linux. En effet, la majorité des programmes malveillants possèdent une extension « .exe ». Or, ceux-ci ne sont pas exécutables sous Linux sans passer par des manipulations. Dans ce contexte, **un virus restera inactif**.

Le deuxième atout est lié au nombre d'utilisateurs du soft Open Source. Moins important que chez la concurrence, il limite indirectement les tentatives d'hacking (depuis un **ransomware** ou un **cheval de Troie**).

Conseils comment se protéger sur Linux :

Pare feu du système

Un pare-feu est initialement installé sur la machine. Toutefois il est préférable de vérifier les paramètres de celui-ci.

Mettre à jour Linux

Il est très important de mettre à jour régulièrement les paquets Linux. Il est important aussi de faire attention à la version des distributions.

Autres conseils :

- Ne pas télécharger ses données sensibles sur le cloud
- Faire des copies de sauvegarde
- Ne pas laissez des données sensibles sur des forums
- Utilisez des mots de passe forts
- Ne pas installer de paquets depuis des sources non connues

Sécurité du mot de passe

MySQL supporte les gestions de mot de passe suivants :

- Expiration du mot de passe, requiert son changement périodiquement
- Restriction des réutilisations, pour empêcher d'anciens mot de passe d'être choisi à nouveau
- Vérification de mot de passe, requiert que pour changer de mot de passe il faut également préciser le mot de passe actuel
- Duo de mot de passe, pour permettre à l'utilisateur de se connecter en utilisant soit un mot de passe primaire soit secondaire.
- Evaluation de mot de passe, pour autoriser des mots de passe forts
- Génération de mot de passe aléatoire, une alternative aux mots de passe spécifiés.
- Suivi d'échec de mot de passe, pour bloquer temporairement un compte après trop de tentatives incorrectes pour se connecter.

À partir de MySQL 8.0.13, il est possible d'exiger que les tentatives de modification d'un mot de passe de compte soient vérifiées en spécifiant le mot de passe actuel à remplacer.

Cela permet aux administrateurs de bases de données d'empêcher les utilisateurs de modifier un mot de passe sans prouver qu'ils connaissent le mot de passe actuel.

De telles modifications pourraient autrement se produire, par exemple, si un utilisateur quitte temporairement une session de terminal sans se déconnecter et qu'un utilisateur malveillant utilise la session pour modifier le mot de passe MySQL de l'utilisateur d'origine.

Quelques plugins d'authentification sont internes à MySQL.

Dans la table système `mysql.user` :

- `mysql_native_password`
- `caching_sha2_password`
- `sha256_password`

D'autres plugins d'authentification externes à MySQL existent. Pour les comptes qui utilisent ces plugins, la gestion du mot de passe doit être pris en compte dans ces systèmes externes également.

Les seules exceptions concernent les logs d'échec d'authentification et la restriction temporaire de compte qui s'appliquent à tous les comptes, pas seulement ceux utilisant des moyens d'authentification internes, puisque MySQL est capable d'accéder au statut des tentatives de connexion de chaque compte peu importe qu'il utilise un moyen d'authentification interne ou externe.

La sécurité du système de fichiers

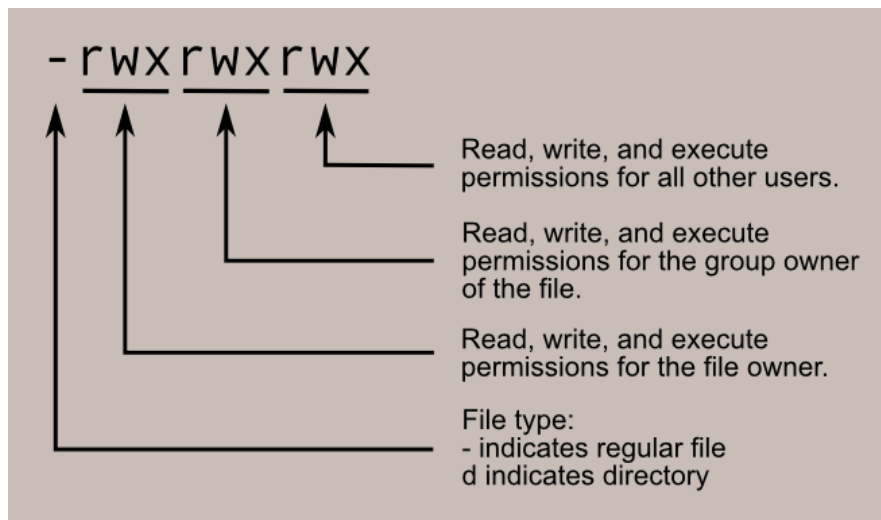
Droits d'accès : la première ligne de défense de Linux

Le modèle de sécurité Linux est basé sur celui utilisé sur les systèmes UNIX. Sur Linux chaque fichier appartient à un utilisateur et à un utilisateur de groupe. Il existe différents groupes d'utilisateurs (propriétaire, visiteur ...)

Pour chaque catégorie d'utilisateurs, des autorisations de lecture, d'écriture et d'exécution peuvent être accordées ou refusées.

Grâce à la commande `ls -l` on peut afficher les autorisations de fichiers pour les catégories d'utilisateurs ; ils sont indiqués par neuf caractères. Un dixième caractère qui se trouve en premier, est l'indicateur de type de fichier au début de la ligne des propriétés du fichier.

Les trois premiers caractères de cette série de neuf affichent des droits d'accès pour l'utilisateur réel propriétaire du fichier. Les trois suivants sont pour le propriétaire du groupe du fichier, les trois derniers pour les autres utilisateurs.



Code d'accès aux fichiers :

Code	Signification
0 ou -	Le droit d'accès qui est censé se trouver à cet endroit n'est pas accordé.
4 ou r	L'accès en lecture est accordé à la catégorie d'utilisateurs définie à cet endroit.
2 ou w	L'autorisation d'écriture est accordée à la catégorie d'utilisateurs définie à cet emplacement.
1 ou x	L'autorisation d'exécution est accordée à la catégorie d'utilisateurs définie à cet emplacement.

Code des groupes utilisateurs :

Code	Signification
U	Permission utilisateur actuel.
G	Permission sur le groupe.
O	Permission pour les autres.

Astuce :

Le nom de l'utilisateur est stocké dans une variable d'environnement : **echo \$USER**

Protection des données de la base

Qu'est-ce que la sécurité de la base de données ?

La sécurité des bases de données sont toutes les mesures que les entreprises prennent pour **prévenir** les violations de leur système de gestion de base de données.

Ce problème n'est pas seulement présent pour les grandes entreprises, les petites organisations doivent également mettre en place des normes de sécurité pour les bases de données.

1. Séparez les serveurs de base de données et les serveurs Web

Garder le serveur de base de données dans un environnement sécurisé et verrouillé avec des contrôles d'accès en place pour empêcher les personnes non autorisées d'entrer.

Un serveur Web est plus susceptible d'être attaqué (accessible au public). Et si un serveur Web est compromis et que le serveur de base de données s'exécute sur la même machine, l'attaquant aurait accès en tant qu'utilisateur root à votre base de données et à vos données.

2. Utilisez des pare-feux d'applications Web et de bases de données

Votre serveur de base de données doit être protégé contre les menaces de sécurité de la base de données par un pare-feu, qui refuse l'accès au trafic par défaut. Le seul trafic autorisé doit provenir d'applications ou de serveurs Web spécifiques qui doivent accéder aux données.

3. Accès utilisateur sécurisé à la base de données

Vous devez viser le moins de personnes possible pour avoir accès à la base de données. Les administrateurs ne doivent disposer que des privilèges minimaux dont ils ont besoin pour faire leur travail.

En plus :

- Des mots de passe forts doivent être appliqués
- Les hachages de mot de passe doivent être stockés cryptés.
- Les comptes doivent être verrouillés après trois ou quatre tentatives de connexion

Les types de droits existants

Nous avons des droits sur les colonnes, les tables, les bases de données et les serveurs.

```
GRANT ALL ON db1.* TO 'jeffrey'@'localhost';
GRANT 'role1', 'role2' TO 'user1'@'localhost', 'user2'@'localhost';
GRANT SELECT ON world.* TO 'role3';
```

```
REVOKE INSERT ON *.* FROM 'jeffrey'@'localhost';
REVOKE 'role1', 'role2' FROM 'user1'@'localhost', 'user2'@'localhost';
REVOKE SELECT ON world.* FROM 'role3';
```

Les droits peuvent être ajoutés avec **GRANT** et retirés avec **REVOKE**.

Il y a des privilèges **statiques** et des privilèges **dynamiques**. Les privilèges statiques sont définis dans le serveur tandis que les privilèges dynamiques sont définis au démarrage.

Privilèges statiques :

Privilege	Grant Table Column	Context
ALL [PRIVILEGES]	Synonym for "all privileges"	Server administration
ALTER	Alter_priv	Tables
ALTER ROUTINE	Alter_routine_priv	Stored routines
CREATE	Create_priv	Databases, tables, or indexes
CREATE ROLE	Create_role_priv	Server administration
CREATE ROUTINE	Create_routine_priv	Stored routines
CREATE TABLESPACE	Create_tablespace_priv	Server administration
CREATE TEMPORARY TABLES	Create_tmp_table_priv	Tables
CREATE USER	Create_user_priv	Server administration
CREATE VIEW	Create_view_priv	Views
DELETE	Delete_priv	Tables
DROP	Drop_priv	Databases, tables, or views
DROP ROLE	Drop_role_priv	Server administration
EVENT	Event_priv	Databases
EXECUTE	Execute_priv	Stored routines
FILE	File_priv	File access on server host
GRANT OPTION	Grant_priv	Databases, tables, or stored routines
INDEX	Index_priv	Tables
INSERT	Insert_priv	Tables or columns
LOCK TABLES	Lock_tables_priv	Databases
PROCESS	Process_priv	Server administration
PROXY	See proxies_priv table	Server administration
REFERENCES	References_priv	Databases or tables
RELOAD	Reload_priv	Server administration
REPLICATION CLIENT	Rep_l_client_priv	Server administration
REPLICATION SLAVE	Rep_l_slave_priv	Server administration
SELECT	Select_priv	Tables or columns
SHOW DATABASES	Show_db_priv	Server administration
SHOW VIEW	Show_view_priv	Views
SHUTDOWN	Shutdown_priv	Server administration
SUPER	Super_priv	Server administration
TRIGGER	Trigger_priv	Tables
UPDATE	Update_priv	Tables or columns
USAGE	Synonym for "no privileges"	Server administration

Privilèges dynamiques :

Privilege	Context
<u>APPLICATION_PASSWORD_ADMIN</u>	Dual password administration
<u>AUDIT_ABORT_EXEMPT</u>	Allow queries blocked by audit log filter
<u>AUDIT_ADMIN</u>	Audit log administration
<u>AUTHENTICATION_POLICY_ADMIN</u>	Authentication administration
<u>BACKUP_ADMIN</u>	Backup administration
<u>BINLOG_ADMIN</u>	Backup and Replication administration
<u>BINLOG_ENCRYPTION_ADMIN</u>	Backup and Replication administration
<u>CLONE_ADMIN</u>	Clone administration
<u>CONNECTION_ADMIN</u>	Server administration
<u>ENCRYPTION_KEY_ADMIN</u>	Server administration
<u>FIREWALL_ADMIN</u>	Firewall administration
<u>FIREWALL_EXEMPT</u>	Firewall administration
<u>FIREWALL_USER</u>	Firewall administration
<u>FLUSH_OPTIMIZER_COSTS</u>	Server administration
<u>FLUSH_STATUS</u>	Server administration
<u>FLUSH_TABLES</u>	Server administration
<u>FLUSH_USER_RESOURCES</u>	Server administration
<u>GROUP_REPLICATION_ADMIN</u>	Replication administration
<u>GROUP_REPLICATION_STREAM</u>	Replication administration
<u>INNODB_REDO_LOG_ARCHIVE</u>	Redo log archiving administration
<u>NDB_STORED_USER</u>	NDB Cluster
<u>PASSWORDLESS_USER_ADMIN</u>	Authentication administration
<u>PERSIST_RO_VARIABLES_ADMIN</u>	Server administration
<u>REPLICATION_APPLIER</u>	PRIVILEGE_CHECKS_USER for a replication channel
<u>REPLICATION_SLAVE_ADMIN</u>	Replication administration
<u>RESOURCE_GROUP_ADMIN</u>	Resource group administration
<u>RESOURCE_GROUP_USER</u>	Resource group administration
<u>ROLE_ADMIN</u>	Server administration
<u>SENSITIVE_VARIABLES_OBSERVER</u>	Server administration
<u>SESSION_VARIABLES_ADMIN</u>	Server administration
<u>SET_USER_ID</u>	Server administration
<u>SHOW_ROUTINE</u>	Server administration
<u>SKIP_QUERY_REWRITE</u>	Server administration
<u>SYSTEM_USER</u>	Server administration
<u>SYSTEM_VARIABLES_ADMIN</u>	Server administration
<u>TABLE_ENCRYPTION_ADMIN</u>	Server administration
<u>TP_CONNECTION_ADMIN</u>	Thread pool administration
<u>VERSION_TOKEN_ADMIN</u>	Server administration
<u>XA_RECOVER_ADMIN</u>	Server administration

Nous avons notamment les droits d'administrateur donné avec **ALL**.

Certaines permissions peuvent être appliqué sur différent objets. On a **DROP** qui permet de supprimer des tables, base et autre. C'est une permission qui peut donc être assignée à une table ou une base de données.

Certaines permissions se rapproche des permissions chmod.

On a **SELECT** qui permet de récupérer une colonne ou une table. SELECT correspond à la permission *read*.

Ensuite on a **UPDATE** qui permet d'éditer une colonne d'une ligne de la table.

On a aussi **INSERT** qui permet d'insérer des ligne à la table. INSERT est requis pour certaine fonctions de maintenances.

Update et insert correspondent à *write* dans la commande *chmod*.

Enfin, on a **DELETE** qui permet d'effacer une ligne d'une table, index pour créer un index sur une table existante.

CREATE permet de créer une table ou une base de données.

La gestion de l'utilisateur lors de la création de routine

MySQL permet de créer des routines.

Une routine est une suite de requêtes SQL stocké sur un serveur.

Elles sont assez pratiques puisqu'elles permettent d'exécuter plus facilement une même requête plusieurs fois.

Les routines permettent aussi de réduire les informations qui transitent entre le serveur et le client.

En contrepartie la charge côté serveur est augmentée puisque plus de tâches sont exécutées sur le serveur.

Voici un exemple de procédure :

```
mysql> CREATE PROCEDURE citycount (IN country CHAR(3), OUT cities INT)
      BEGIN
        SELECT COUNT(*) INTO cities FROM world.city
        WHERE CountryCode = country;
      END//
Query OK, 0 rows affected (0.01 sec)
```

On doit utiliser la commande CREATE PROCEDURE qui permet de définir la procédure, ici *citycount*.

Entre les parenthèses on doit mettre IN suivi de la variable qui va stocker l'argument que nous allons donner et OUT suivi de celui qu'on doit recevoir.

Ensuite on doit dire qu'on commence la requête SQL avec BEGIN.

On met notre requête et on met END pour dire que la requête est terminée :

```
mysql> CALL citycount('JPN', @cities); -- cities in Japan
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @cities;
+-----+
| @cities |
+-----+
|      248 |
+-----+
1 row in set (0.00 sec)
```

Pour appeler la procédure il suffit d'utiliser CALL + le nom de la procédure.

Puis entre les parenthèses il faut mettre l'argument qu'on veut passer et la variable qui va récupérer la réponse.

Pour afficher le résultat il faut utiliser SELECT suivi de la variable.

Pour les fonctions c'est un peu différent. Ici nous avons la création de la fonction hello grâce à CREATE FUNCTION :

```
mysql> CREATE FUNCTION hello (s CHAR(20))
mysql> RETURNS CHAR(50) DETERMINISTIC
      RETURN CONCAT('Hello, ',s,'!');
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT hello('world');
+-----+
| hello('world') |
+-----+
| Hello, world!  |
+-----+
1 row in set (0.00 sec)
```

Entre les parenthèses il faut mettre le ou les arguments que nous voulons passer.

Ensuite il faut mettre un RETURN avec l'opération que nous voulons faire.

L'utilisation de RETURNS permet de faire plusieurs return dans une même fonction. Comme nous pouvons le voir ici grâce à SELECT puis la fonction.

Création, suppression, modification des utilisateurs

- 1- Un utilisateur est juste composé à minima d'un pseudo, d'un num d'utilisateur, dans certains lieux il peut s'agir du prénom quand il y a plusieurs personnes on peut rajouter (ou le faire de manière automatique pour ne pas être confronté au problème) et rajouter un host qui précise l'identité de la personne (exemple l'IP du post).
- 2- Pour sécuriser l'authentification nous pouvons rajouter une méthode d'authentification, soit par mot de passe qui est gérée par MySQL soit par un plugin pour une authentification externe.
- 3- Il est plutôt simple de supprimer un utilisateur, on peut aussi le modifier par requêtes SQL sur la table d'utilisateur, ou alors suivant la version de SQL modifier par ligne de commande.

Gestion des privilèges et des rôles

- 1- Le compte d'un utilisateur pourra se voir accordé des rôles qui auront un ou plusieurs privilèges rattachés. (*expliquer schéma*)
- 2- Il est plus pratique de gérer des rôles qui sont définis par des privilèges, le rôle régit une fonctionnalité disponible.
- 3- Il est du coup plus pratique d'attribuer un ou plusieurs rôles qui sont définis par une liste de privilèges que de les accorder directement aux utilisateurs.
- 4- Les rôles *mandatory* sont des rôles qui sont jugés innés à chaque utilisateur, ils n'ont pas besoin de lui être attribués.
Ils ne sont pas présents dans la liste des rôles de l'utilisateur vu que la liste est gérée sur le serveur. Une modification de cette liste nécessite le redémarrage de celui-ci.
- 5- Pour que les utilisateurs aient accès aux rôles qui leur ont été attribués après leurs connexions il faut définir la ou les rôles qui vont lui être activés par défaut, il est possible de désactiver un ou plusieurs rôles alors qu'ils sont activés. (Sanction ou attribution temporaire, etc)

- 6- La suppression d'un rôle est très simple, plus de précisions sont possibles pour retirer plus de rôles ou spécifier sur quelle base de donnée ou table appliquer cela.
- 7- Dans les anciennes versions de MySQL tout changements de rôle, privilège et de leurs attributions nécessite de sauvegarder les changements pour les mettre en pratique ! Cette commande n'est plus nécessaire dans les nouvelles version mais a été garder dans la documentation pour les vieilles version.

La sécurité du réseau

Nous allons voir comment configurer MySQL pour permettre les connexions à distance du serveur d'application. Si notre base de données nécessite un accès à d'autres serveurs, il y a quelques étapes à respecter pour sécuriser MySQL.

Les outils disponibles

MySQL fournit plusieurs outils pour la sécurité du réseau. Notamment :

- **MySQL Bind-Address** : comment et depuis quelles adresses MySQL écoute les connexions.
- **La sécurité des utilisateurs** : qui peut se connecter, et depuis où.
- **Parefeu** : quel trafic le serveur accepte ou rejette.

MySQL Bind Address

La configuration du **bind-address** dans MySQL définit sur quel réseau MySQL peut écouter pour établir une connexion.

MySQL est habituellement configuré pour accepter les connexions depuis un fichier socket local (une socket Unix). Les paramètres de **bind-address** vont dire à MySQL s'il est autorisé ou non à se brancher sur une socket TCP pour établir une connexion. Il y a 3 configurations principales de **bind-address** :

- MySQL ne peut se brancher sur aucun réseau : seule une connexion via localhost est possible (socket Unix).
- MySQL peut se brancher sur tous les réseaux (0.0.0.0).
- MySQL peut se brancher sur un réseau spécifique, c-à-d soit un réseau public accessible par tout le monde, soit un réseau privé accessible uniquement depuis un data center.

La configuration par défaut du **bind-address** est d'écouter tous les réseaux. C'est un paramètre qui n'est pas sécurisé. Plus on sera restrictif, mieux ce sera.

Si notre application est sur le même serveur que la base de données, on peut empêcher MySQL de se brancher sur d'autres réseaux (et donc de n'autoriser qu'une connexion via la socket Unix locale).

Le plus courant est d'autoriser également l'adresse de loopback 127.0.0.1 pour qu'une connexion soit possible à la fois via le **localhost** (la socket Unix) ET l'adresse **127.0.0.1** (donc la socket TCP), mais rien d'autre.

La configuration ressemble alors à ceci :

```
[mysqld]
# Unix socket settings (making localhost work)
user          = mysql
pid-file      = /var/run/mysqld/mysqld.pid
socket        = /var/run/mysqld/mysqld.sock

# TCP Socket settings (making 127.0.0.1 work)
port          = 3306
bind-address  = 127.0.0.1
```

La sécurité des utilisateurs MySQL

En plus de configurer les réseaux sur lesquels MySQL peut se brancher, on peut définir depuis quelle adresse les utilisateurs peuvent se connecter. Par exemple, « l'utilisateur toto peut uniquement se connecter à MySQL depuis le serveur dont l'adresse est 192.168.33.10 ».

On commence avec la commande suivante pour afficher la liste des utilisateurs existant sur le serveur MySQL :

```
mysql> SELECT User, Host from mysql.user;
```

```
+-----+-----+
| User      | Host      |
+-----+-----+
| root      | 127.0.0.1 |
| root      | ::1       |
| mysql.sys | localhost |
| root      | localhost |
+-----+-----+
4 rows in set (0.00 sec)
```

Par défaut, on voit qu'on a 3 utilisateurs root et 1 utilisateur système.

- root@127.0.0.1 : peut se connecter via l'adresse de loopback 127.0.0.1
- root@::1 : peut se connecter via le réseau de loopback IPv6 ::1
- root@localhost : peut se connecter via la socket Unix.

Utilisateurs applicatifs

Il faut maintenant créer des utilisateurs MySQL.

Imaginons que notre serveur MySQL soit hébergé sur le réseau suivant :

- Adresse IPv4 publique : 159.203.81.145
- Adresse IPv4 privée : 10.132.30.23

...et qu'un serveur applicatif soit dans le même data center sur le réseau :

- Adresse IPv4 publique : 104.131.100.163
- Adresse IPv4 privée : 10.132.51.34

Comme les deux serveurs sont dans le même réseau privé en 10.132, ils peuvent communiquer entre eux.

Configurons le serveur applicatif pour qu'il puisse se connecter au serveur MySQL. On peut utiliser plusieurs outils pour ça :

- Le nom d'hôte (*example.com*)
- Les adresses IP spécifiques (192.168.10.10)
- Les wildcards (192.168.10.%)
- Le masque de sous-réseau

```
-- Depuis le serveur MySQL, créer de nouveaux utilisateurs
```

```
-- Créer un utilisateur qui peut uniquement se connecter depuis ce serveur  
CREATE USER 'toto'@'10.132.51.34' IDENTIFIED BY 'mot-de-passe';
```

```
-- Créer un utilisateur qui peut se connecter depuis n'importe quelle IP  
commençant par 10.132.51  
CREATE USER 'toto'@'10.132.51.%' IDENTIFIED BY 'mot-de-passe';
```

```
-- Créer un utilisateur qui peut se connecter depuis n'importe quelle IP
commençant par 10.132
CREATE USER 'toto'@'10.132.%' IDENTIFIED BY 'mot-de-passe;

-- Créer un utilisateur qui peut se connecter depuis n'importe quelle IP
commençant par 10
CREATE USER 'toto'@'10.%' IDENTIFIED BY 'mot-de-passe;

-- Créer un utilisateur qui peut se connecter depuis 10.132.51.*
CREATE USER 'toto'@'10.132.51.0/255.255.255.0' IDENTIFIED BY 'mot-de-passe;

-- Créer un utilisateur qui peut se connecter depuis 10.132.*.*
CREATE USER 'toto'@'10.132.0.0/255.255.0.0' IDENTIFIED BY 'mot-de-passe;

-- Créer un utilisateur qui peut se connecter depuis 10.*.*.*
CREATE USER 'toto'@'10.0.0.0/255.0.0.0' IDENTIFIED BY 'mot-de-passe;
```

Le premier exemple est très spécifique : l'utilisateur ne peut se connecter à MySQL que s'il se connecte à ce serveur depuis l'adresse IP 10.132.51.34.

Les autres exemples utilisent les wildcards (%). Les wildcards reposent sur le même principe que le mot clé LIKE en SQL, et sont souvent utilisées pour la correspondance des noms d'hôtes : 'toto'@'%example.com'

Enfin, les derniers exemples reposent sur les masques de sous-réseaux, qui sont spécifiques aux adresses IPv4.

Les réseaux publics

Le même principe s'applique aux réseaux publics. Si notre base de données a besoin d'être accessible depuis d'autres régions du monde, alors le réseau privé ne fonctionnera plus. Dans ce cas, il serait recommandé de mettre en place un VPN ou un tunnel SSL, pour qu'un réseau privé puisse être configuré sur plusieurs régions.

Néanmoins, les VPN restent assez complexes à paramétrer et à maintenir, et on peut s'en passer.

On peut brancher notre MySQL soit sur tous les réseaux (donc 0.0.0.0) pour que l'accès au réseau privé soit disponible, soit directement au réseau public (dans notre exemple : 159.203.81.145).

La configuration de l'utilisateur est donc identique :

```
-- Créer un utilisateur qui peut uniquement se connecter depuis ce serveur
CREATE USER 'toto'@'104.131.100.163' IDENTIFIED BY 'mot-de-passe;
```

Parefeu

En plus des outils proposés par MySQL, il est conseillé d'utiliser un pare-feu pour mieux se protéger. Par exemple, on peut paramétrer notre pare-feu pour qu'il n'accepte que les connexions au port 3306 si elles arrivent sur notre réseau privé.

```
# Append rule to the end of the INPUT chain
sudo iptables -A INPUT -p tcp -m tcp --dport 3306 -d
10.132.30.23 -j ACCEPT

# Or, insert it into the middle of our chain (position 3 here)
sudo iptables -I INPUT 3 -p tcp -m tcp --dport 3306 -d
10.132.30.23 -j ACCEPT
```

Le flag **-d** définit le réseau de destination.

On peut aussi configurer quelle interface réseau autorise le trafic sur le port 3306 :

```
# Append rule to the end of the INPUT chain
sudo iptables -A INPUT -p tcp -m tcp --dport 3306 -i eth1 -j ACCEPT

# Or, insert it into the middle of our chain (position 3 here)
sudo iptables -I INPUT 3 -p tcp -m tcp --dport 3306 -i eth1 -j ACCEPT
```

On peut également configurer des règles pour autoriser le trafic depuis des adresses IP spécifiques ou des réseaux spécifiques, en utilisant les masques de sous-réseaux ou la notation CIDR :

```
# Allow traffic from a specific IP
sudo iptables -A INPUT -p tcp -m tcp --dport 3306 -s 10.132.51.34 -j
ACCEPT

# Allow traffic from a netmask range of IP addresses 10.132.51.*
sudo iptables -A INPUT -p tcp -m tcp --dport 3306 -s
10.132.51.0/255.255.255.0 -j ACCEPT

# Allow traffic from a CIDR range of IP addresses 10.132.51.*
sudo iptables -A INPUT -p tcp -m tcp --dport 3306 -s 10.132.51.0/24 -j
ACCEPT
```

TL ;DR

1. Branchez MySQL sur l'adresse 0.0.0.0 si vous voulez pouvoir vous brancher sur tous les réseaux, privés comme publics. Branchez-vous sur l'adresse IP du réseau privé de MySQL pour ne vous brancher que sur celui-ci.
2. En cas de connexion au localhost, MySQL se branchera sur une socket Unix locale
3. Configurez de nouveaux utilisateurs MySQL et restreignez les adresses depuis lesquelles ils peuvent se connecter (une adresse spécifique ou un intervalle d'IP aussi petit que possible)
4. Utilisez un pare-feu en conjonction avec les restrictions utilisateur de MySQL.

Connection Over SSL

SSL → Secure Sockets Layer

TLS → Transport Layer Security (Successeur)

Protocoles de sécurisation des échanges par réseau informatiques (internet)

Fonctionnement client-serveur, sécurisation par :

- Authentification du serveur
- Confidentialité des Données échangées (session chiffrée)
- Intégrité des données échangées
- Authentification du client souvent assurée par la couche applicative (Optionnel)

Mise en oeuvre facilitée car juste à "ajouter" le SSL/TLS pour passer de HTTP à HTTPS

mTLS → mutual TLS

TLS + Authentification obligatoire du client

Type de connexion sécurisée :

- Utilisateur + mot de passe hashé
- `unix_socket` (plugin) => connexion basée sur les droits de la session (linux)

TCP + transport.

Trames.

Login via session user `unix_socket` plugin.

Authentication externe

Il y a différents moyens de se connecter via une authentification externe. Nous allons surtout regarder la manière avec Socket.

Pour commencer, on a deux possibilités :

1. La première possibilité est de modifier le fichier de configuration my.cnf et d'ajouter à la ligne : «plugin-load-add» auth_socket.so

```
[mysqld]  
plugin-load-add=auth_socket.so
```

2. La deuxième est d'utiliser la commande «INSTALL PLUGIN auth_socket SONAME 'auth_socket.so'»

```
INSTALL PLUGIN auth_socket SONAME 'auth_socket.so';
```

Cela permettra de créer des utilisateurs via leur session sur leur ordinateur grâce à l'IP et le nom de session.

```
CREATE USER 'valerie'@'localhost' IDENTIFIED WITH auth_socket;
```

Sources :

- **Injections SQL :**
Microsoft.com :
<https://learn.microsoft.com/fr-fr/sql/relational-databases/security/sql-injection?view=sql-server-ver16>
- **Installation de MySQL :**
Gilad David Maayan sur **Computer.org** :
<https://www.computer.org/publications/tech-news/trends/mysql-security-best-practices>
- **Sécurité de l'OS :**
Aaron Sampson chez Skillsoft :
https://www.youtube.com/watch?v=9_Bpxbkwjvw
- **Sécurité du système de fichiers :**
 - o <https://www.linuxtopia.org/>
 - o <https://tldp.org/>
- **Sécurité du mot de passe :**
Mysql.com :
<https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/password-management.html>
- **Types de droits existants :**
Mysql.com :
 - o <https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html>
 - o <https://dev.mysql.com/doc/refman/8.0/en/grant.html>
 - o <https://dev.mysql.com/doc/refman/8.0/en/revoke.html>
- **Création, suppression, modification des utilisateurs, gestion des privilèges et des rôles :**
 - o <https://serverfault.com/questions/483339/changing-host-permissions-for-mysql-users>
 - o <https://www.prisma.io/dataguide/mysql/short-guides/creating-deleting-users>
 - o <https://dev.mysql.com/doc/refman/8.0/en/roles.html>
 - o <https://stackoverflow.com/questions/50969509/user-role-permissions-and-a-specific-group-rbac>

- **Connection Over SSL :**

Mysql.com :

<https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html>

Medium.com :

<https://medium.com/double-pointer/ssl-vs-tls-vs-mtls-f5e836fe6b6d>

- **Sécurité du réseau :**

Serversforhackers.com :

<https://serversforhackers.com/c/mysql-network-security>